

Docket No. 205127US2

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: Kazuyoshi KOHNO, et al.

GAU:

SERIAL NO: New Application

EXAMINER:

FILED: Herewith

FOR: DESIGN VERIFICATION METHOD, DESIGN VERIFICATION DEVICE FOR MICROPROCESSORS, AND PIPELINE SIMULATOR GENERATION DEVICE

REQUEST FOR PRIORITY

ASSISTANT COMMISSIONER FOR PATENTS
WASHINGTON, D.C. 20231

SIR:

- ☐ Full benefit of the filing date of U.S. Application Serial Number, filed, is claimed pursuant to the provisions of 35 U.S.C. §120.
- ☐ Full benefit of the filing date of U.S. Provisional Application Serial Number, filed, is claimed pursuant to the provisions of 35 U.S.C. §119(e).
- ☒ Applicants claim any right to priority from any earlier filed applications to which they may be entitled pursuant to the provisions of 35 U.S.C. §119, as noted below.

In the matter of the above-identified application for patent, notice is hereby given that the applicants claim as priority:


<u>COUNTRY</u>	<u>APPLICATION NUMBER</u>	<u>MONTH/DAY/YEAR</u>
JAPAN	2000-087411	March 27, 2000

Certified copies of the corresponding Convention Application(s)

- ☒ are submitted herewith
- ☐ will be submitted prior to payment of the Final Fee
- ☐ were filed in prior application Serial No. filed
- ☐ were submitted to the International Bureau in PCT Application Number .
Receipt of the certified copies by the International Bureau in a timely manner under PCT Rule 17.1(a) has been acknowledged as evidenced by the attached PCT/IB/304.
- ☐ (A) Application Serial No.(s) were filed in prior application Serial No. filed ; and
(B) Application Serial No.(s)
- ☐ are submitted herewith
- ☐ will be submitted prior to payment of the Final Fee

Respectfully Submitted,

OBLON, SPIVAK, McCLELLAND,
MAIER & NEUSTADT, P.C.


Marvin J. Spivak
Registration No. 24,913



22850

Tel. (703) 413-3000
Fax. (703) 413-2220
(OSMMN 10/98)



日本国特許庁
PATENT OFFICE
JAPANESE GOVERNMENT

655
J0974 U.S. PTO
09/816480
03/26/81

別紙添付の書類に記載されている事項は下記の出願書類に記載されて
いる事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed
with this Office.

出願年月日
Date of Application:

2000年 3月27日

出願番号
Application Number:

特願2000-087411

出願人
Applicant(s):

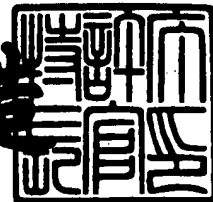
株式会社東芝

CERTIFIED COPY OF
PRIORITY DOCUMENT

2000年12月22日

特許庁長官
Commissioner,
Patent Office

及川耕造



出証番号 出証特2000-3105750

【書類名】 特許願

【整理番号】 4HA99Z073

【提出日】 平成12年 3月27日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 7/00

【発明の名称】 マイクロプロセッサの設計検証方法及びマイクロプロセッサの設計検証装置及びパイプラインシミュレータ生成装置

【請求項の数】 3

【発明者】

【住所又は居所】 神奈川県川崎市幸区小向東芝町1番地 株式会社東芝
マイクロエレクトロニクスセンター内

【氏名】 河野 和義

【発明者】

【住所又は居所】 神奈川県川崎市幸区小向東芝町1番地 株式会社東芝
マイクロエレクトロニクスセンター内

【氏名】 水野 淳

【特許出願人】

【識別番号】 000003078

【氏名又は名称】 株式会社 東芝

【代理人】

【識別番号】 100083806

【弁理士】

【氏名又は名称】 三好 秀和

【電話番号】 03-3504-3075

【選任した代理人】

【識別番号】 100068342

【弁理士】

【氏名又は名称】 三好 保男

【選任した代理人】

【識別番号】 100100712

【弁理士】

【氏名又は名称】 岩▲崎▼ 幸邦

【選任した代理人】

【識別番号】 100100929

【弁理士】

【氏名又は名称】 川又 澄雄

【選任した代理人】

【識別番号】 100108707

【弁理士】

【氏名又は名称】 中村 友之

【選任した代理人】

【識別番号】 100095500

【弁理士】

【氏名又は名称】 伊藤 正和

【選任した代理人】

【識別番号】 100101247

【弁理士】

【氏名又は名称】 高橋 俊一

【選任した代理人】

【識別番号】 100098327

【弁理士】

【氏名又は名称】 高松 俊雄

【手数料の表示】

【予納台帳番号】 001982

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1
【物件名】 要約書 1
【ブルーフの要否】 要

【書類名】 明細書

【発明の名称】 マイクロプロセッサの設計検証方法及びマイクロプロセッサの設計検証装置及びパイプラインシミュレータ生成装置

【特許請求の範囲】

【請求項 1】 計算機が解釈可能に記述されたマイクロプロセッサのパイプライン仕様に基づいて前記計算機により検証プログラムを生成し、生成された検証プログラムと前記マイクロプロセッサの R T L 記述に基づいて前記 R T L 記述のシミュレーションを実行し、

また、前記パイプライン仕様に基づいて前記計算機によりパイプラインシミュレータを生成し、前記検証プログラムと生成されたパイプラインシミュレータに基づいてパイプラインシミュレーションを実行し、

前記実行された R T L 記述のシミュレーション結果と前記パイプラインシミュレーション結果との比較を行い、比較結果に基づいてマイクロプロセッサのパイプライン動作を検証することを特徴とするマイクロプロセッサの設計検証方法。

【請求項 2】 計算機が解釈可能に記述されたマイクロプロセッサのパイプライン仕様を入力する入力手段と、

前記入力手段により入力された前記パイプライン仕様に基づいて、前記計算機によりパイプラインシミュレータを生成するシミュレータ生成手段と、

前記入力手段により入力された前記パイプライン仕様に基づいて、前記計算機により検証プログラムを生成するプログラム生成手段と、

前記プログラム生成手段により生成された検証プログラムと前記マイクロプロセッサの R T L 記述に基づいて、R T L 記述のシミュレーションを実行する R T L シミュレーション実行手段と、

前記プログラム生成手段により生成された検証プログラムと前記シミュレータ生成手段により生成されたパイプラインシミュレータに基づいて、パイプラインシミュレーションを実行するパイプラインシミュレーション実行手段と、

前記 R T L シミュレーション実行手段により実行されたシミュレーションの結果と前記パイプラインシミュレーション実行手段により実行されたシミュレーションの結果を比較し、比較結果に基づいてマイクロプロセッサのパイプライン動

作を検証する比較手段と

を有することを特徴とするマイクロプロセッサの設計検証装置。

【請求項 3】 計算機が解釈可能に記述されたマイクロプロセッサのパイプライン仕様を入力する入力手段と、

前記入力手段により入力された前記パイプライン仕様に基づいて、前記計算機によりパイプラインシミュレータを生成するシミュレータ生成手段とを有することを特徴とするパイプラインシミュレータ生成装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、マイクロプロセッサのRTL記述の論理検証を行なうマイクロプロセッサの設計検証方法及びマイクロプロセッサの設計検証装置及びパイプラインシミュレータ生成装置に関する。

【0002】

【従来の技術】

従来、この種の検証を行う方法としては、例えば図3のフローチャートに示す様な手順が知られていた。図3において、破線で示す部分は、プログラムに基づいた計算機による自動生成ではなく、人手によってなされる処理である。日本語や英語等の自然言語で記述されたパイプライン仕様に基づいて検証しようとするプロセッサのRTL（レジスタ転送レベル）記述を手手により作成する（ステップS101）。また、パイプライン仕様からシミュレータのソースプログラムを手手により作成し（ステップS102）、作成されたソースプログラムとパイプライン仕様には依存しないシミュレータのソースプログラム（ステップS103）とからパイプライン・シミュレータを生成する（ステップS104）。さらに、パイプライン仕様から検証項目を手手により作成し（ステップS105）、検証項目から複数の検証プログラムを作成する（ステップS106）。

【0003】

次に、プロセッサのRTL記述と検証プログラムとからRTLシミュレータによりRTLシミュレーションを行い（ステップS107）、また検証プログラム

を用いてパイプラインシミュレータによりパイプライン・シミュレーションを行い（ステップ S 1 0 8）、RTLシミュレーションの結果（ステップ S 1 0 9）とパイプライン・シミュレーションの結果（ステップ S 1 1 0）とを比較し（ステップ S 1 1 1）、パイプライン動作の正否（PASS/FAIL）を判定する（ステップ S 1 1 2）。

【 0 0 0 4 】

このような検証手順においては、以下に説明するような不具合を招いていた。この不具合の多くはパイプライン仕様が曖昧であることに起因していた。上記従来の検証方法において、パイプライン仕様は日本語、英語等の作業者が容易に理解できる自然言語で記述されていた。このため、本来厳密に定義されるべきパイプライン仕様が、仕様を記述する人間の能力に左右されることになっていた。このような状況は、自然言語で記述されたパイプライン仕様を解釈する場合にも当てはまる。このことにより、RTL実装者、パイプライン・シミュレータ実装者、検証プログラム作成者にパイプライン仕様の誤解が生ずる可能性があった。

【 0 0 0 5 】

RTL検証を行なう場合に、パイプライン・シミュレータ実装者のパイプライン仕様の誤解は検証効率に大きな影響を及ぼす。なぜならば、パイプライン・シミュレータはリファレンス・モデルとして使用されるものであり、RTLでのシミュレーション結果と比較する際に絶対に正しい結果を出すものであることを前提に検証作業が行なわれるからである。

【 0 0 0 6 】

検証プログラム作成者のパイプライン仕様の誤解は、論理検証洩れの問題をはらんでいる。論理検証洩れのままマイクロプロセッサを製造し、実機を用いたアプリケーション・ソフトでの論理検証で誤りが発見された場合には、最悪の場合に、論理設計からやり直す必要が生ずる。また、マイクロアーキテクチャの変更在即座に対応できないことも問題である。1個の命令セットアーキテクチャに対して、様々なマイクロアーキテクチャが存在し、さらにマイクロアーキテクチャの仕様は設計段階で変更されることがある。このため、従来の方法では、マイクロアーキテクチャの仕様が設計段階で変更される度に、パイプライン・シミュレ

ータおよび検証プログラムを人手で変更する必要があった。

【 0 0 0 7 】

【発明が解決しようとする課題】

以上説明したように、マイクロプロセッサのRTL記述における従来の論理検証方法にあつては、自然言語で記述された同一のパイプライン仕様に基づいて、RTL実装、パイプラインシミュレータ実装、検証プログラム作成が行なわれていたため、パイプライン仕様の解釈の食い違いによる誤りが発生していた。さらに、検証プログラムはパイプライン仕様から人手により生成されていたため、検証洩れが起こりやすかった。また、RTLシミュレーションの結果と検証プログラムに基づくパイプラインライン・シミュレーションの結果が不一致の場合に、いずれに不具合があるのかの究明が困難であつた。さらに、プロセッサの性能向上の目的等でマイクロアーキテクチャ仕様の変更が行なわれた場合には、容易に対応することが困難であつた。

【 0 0 0 8 】

そこで、この発明は、上記に鑑みてなされたものであり、その目的とするところは、パイプライン動作を正確かつ十分に検証できるマイクロプロセッサの設計検証方法及びマイクロプロセッサの設計検証装置及びパイプラインシミュレータ生成装置を提供することにある。

【 0 0 0 9 】

【課題を解決するための手段】

上記目的を達成するために、課題を解決する第1の手段は、計算機が解釈可能に記述されたマイクロプロセッサのパイプライン仕様に基づいて前記計算機により検証プログラムを生成し、生成された検証プログラムと前記マイクロプロセッサのRTL記述に基づいて前記RTL記述のシミュレーションを実行し、また、前記パイプライン仕様に基づいて前記計算機によりパイプラインシミュレータを生成し、前記検証プログラムと生成されたパイプラインシミュレータに基づいてパイプラインシミュレーションを実行し、前記実行されたRTL記述のシミュレーション結果と前記パイプラインシミュレーション結果との比較を行い、比較結果に基づいてマイクロプロセッサのパイプライン動作を検証することを特徴とす

る。

【0010】

第2の手段は、計算機が解釈可能に記述されたマイクロプロセッサのパイプライン仕様を入力する入力手段と、前記入力手段により入力された前記パイプライン仕様に基づいて、前記計算機によりパイプラインシミュレータを生成するシミュレータ生成手段と、前記入力手段により入力された前記パイプライン仕様に基づいて、前記計算機により検証プログラムを生成するプログラム生成手段と、

前記プログラム生成手段により生成された検証プログラムと前記マイクロプロセッサのRTL記述に基づいて、RTL記述のシミュレーションを実行するRTLシミュレーション実行手段と、前記プログラム生成手段により生成された検証プログラムと前記シミュレータ生成手段により生成されたパイプラインシミュレータに基づいて、パイプラインシミュレーションを実行するパイプラインシミュレーション実行手段と、前記RTLシミュレーション実行手段により実行されたシミュレーションの結果と前記パイプラインシミュレーション実行手段により実行されたシミュレーションの結果を比較し、比較結果に基づいてマイクロプロセッサのパイプライン動作を検証する比較手段とを有することを特徴とする。

【0011】

第3の手段は、計算機が解釈可能に記述されたマイクロプロセッサのパイプライン仕様を入力する入力手段と、前記入力手段により入力された前記パイプライン仕様に基づいて、前記計算機によりパイプラインシミュレータを生成するシミュレータ生成手段とを有することを特徴とする。

【0012】

【発明の実施の形態】

以下、図面を用いてこの発明の実施形態を説明する。

【0013】

図1はこの発明の一実施形態に係るマイクロプロセッサの設計検証方法の手順を示すフローチャートである。本発明は、マイクロプロセッサのパイプライン仕様記述を入力し、その仕様を用いてサイクル・アキュレートなシミュレータ（以下、パイプライン・シミュレータと記す）を生成する機能を有し、また、入力さ

れたパイプライン仕様を用いて、パイプライン動作モデルを生成する機能を有し、さらに、生成されたパイプライン動作モデルを用いて、パイプライン動作を検証するための必要かつ十分な検証プログラムを生成する機能を有する。このような機能を有することにより、生成された検証プログラムを用いてマイクロプロセッサのRTL記述のシミュレーションを行なうと同時に同一の検証プログラムを用いてパイプライン・シミュレータを実行し、両者の結果を機械的に比較することにより洩れなく効率よくマイクロプロセッサのRTL記述の論理検証を行なうようにしている。

【0014】

まず、図1を参照して、この発明の一実施形態に係るマイクロプロセッサの設計検証方法について説明する。図1の破線の部分は、図3と同様に人手によってなされる処理を示す。図1において、自然言語ではなく、計算機で解釈可能に記述されたパイプライン仕様（詳細は後述する）に基づいて検証しようとするプロセッサのRTL記述を手により作成する（ステップS1）。また、上記パイプライン仕様に基づいて計算機によりシミュレータのソースプログラムを生成し（ステップS2）、生成されたシミュレータのソースプログラムと、パイプライン仕様に依存しないシミュレータのソースプログラム（ステップS3）とからパイプライン・シミュレータを生成する（ステップS4）。さらに、パイプライン仕様から計算機によりパイプラインライン動作モデルを生成し（ステップS5）、生成されたパイプライン動作モデル（ステップS6）に基づいて計算機により検証項目を生成し（ステップS7）、生成された検証項目（ステップS8）とステップS6のパイプライン動作モデルとに基づいて検証プログラムを計算機により生成し（ステップS9）、検証プログラム（ステップS10）を得る。

【0015】

パイプライン仕様記述からパイプライン動作モデルを生成し、更にパイプライン動作モデルを用いて検証プログラムを生成する方法に関しては、例えば、特願平11-74118号に開示されている技術を使用することにより実現可能である。パイプライン動作モデルの生成ステップS5では、パイプラインを構成する各ステージの接続関係を有向グラフで表現したパイプライン構造グラフの作成、

・ 構造ハザードによるストール条件の決定、データ・ハザードによるストール条件の決定、各パイプライン・ステージの命令の組み合わせを状態とみなした有限状態機械 (finite state machine) の生成が行われる。検証項目は、各パイプライン・ステージの状態の組み合わせで表現され、検証項目生成ステップ S 7 では、構造ハザードが起こる各パイプライン・ステージの状態、データ・ハザードが起こる各パイプライン・ステージの状態を列挙して検証項目を生成する。検証プログラム生成ステップ S 9 では、パイプライン動作モデルおよび検証項目から検証項目の各要素を検証するための命令列 (検証プログラム) を自動生成する。これらを用いた命令列の自動生成は、BDD を用いて容易に実現可能である。BDD (Binary Decision Diagram) とは、論理関数のグラフ表現の一種であり、論理関数をコンパクトに表現することができる。前述の有限状態機械を BDD を用いて表現し、初期状態から到達可能な状態集合を計算することにより命令列を生成する。

【 0 0 1 6 】

次に、プロセッサのRTL記述と検証プログラム (ステップ S 1 0) とからRTLシミュレータによりRTLシミュレーションを行い (ステップ S 1 1)、また検証プログラムに基づいてパイプラインラインシミュレータによりパイプライン・シミュレーションを行い (ステップ S 1 2)、RTLシミュレーションの結果 (ステップ S 1 3) とパイプライン・シミュレーションの結果 (ステップ S 1 4) とを比較し (ステップ S 1 5)、パイプライン動作の正否 (PASS/FAIL) を判定する (ステップ S 1 6)。

【 0 0 1 7 】

このようなパイプライン動作の検証手順において、パイプラインの仕様を作成する際に、パイプライン仕様を記述する記述言語を定義し、その定義に従ってパイプライン仕様を記述する。

【 0 0 1 8 】

次に、パイプライン仕様の具体的な記述方法について説明する。本パイプライン仕様には暗黙の制御ルールがあり、特に指定しなくても命令がストールする場合があるものとする。暗黙の制御ルールは次の2つである。

【0019】

(1) 同一名ステージに同時に複数の命令は存在できない。

【0020】

(2) In-Order (インオーダー) で命令が完了する。

【0021】

同一名のステージは、物理的にも同一のハードウェアを意味する。あるステージで命令がストールする場合には、それに続く命令もストールするという制御は、この制御ルールにより実現される。ステージの機能が同じでも、対象となるリソースが異なるため、同時に実行可能なステージがある場合には、パイプライン仕様の記述では異なるステージ名をつけないなければならない。

【0022】

以下では、代表的な命令に対してこの実施形態のパイプライン仕様の記述を例示する。以下に説明するパイプライン仕様の記述において、パイプラインの各ステージは、時間的な流れの順に、Fステージ (Fetch stage: 命令フェッチステージ)、Dステージ (Decode stage: 命令デコードステージ)、Eステージ (Execution stage: 命令実行ステージ)、Mステージ (Memory stage: メモリへの書き込み/読み出しステージ)、Wステージ (Write back stage: レジスタへの書き込みステージ) とする。まず、ALU命令のパイプライン仕様記述の一例を示す。バイパスの動作は次のように記述する。例えば、ALUパイプラインでは、Wステージで計算結果をレジスタへ書き込むが、バイパスによりEステージおよびMステージの直後で値が参照可能であるとした場合は、次のように記述する。

【0023】

ALU. Access (E) ;

ALU. Forwarding (E+M) ;

REG. WriteBack (W) ;

一般的には次のような記述となる。

【0024】

演算ユニット. Access (ステージ (の並び))

演算ユニット. Forwarding (ステージ (の並び))

レジスタ. Read (ステージ)

レジスタ. WriteBack (ステージ)

Access はリソース競合判定のために使われる。この指定によりリソース競合によるストールが実行される。Read はオペランドが揃うタイミングの判断基準となる。シミュレータ内部ではオペランドがすべてそろったステージで命令を実際に実行する。Read が指定されていない命令では、発行ステージ (現在は D) の直後のステージで命令を実行している。Forwarding や WriteBack の指定はデータ利用可能の判定で使用する。判定アルゴリズムを以下に示す。

【0025】

(データ利用可能判定アルゴリズムの一例)

if (Forwarding が定義されている) then

Forwarding

並びの先頭ステージが現サイクルで実行終了、またはすでに実行終了なら利用可能。

【0026】

else if (WriteBack が定義されている) then

WriteBack ステージが現サイクルで実行終了、またはすでに実行終了なら利用可能

else 利用不可能

endif

次に、ストール動作は以下のように記述される。例えば汎用レジスタ REG に対して RAW ハザードを検出した場合に、D ステージでストールするパイプライン動作を定義する場合は、次のように記述する。

【0027】

StallStage (RAW, REG) = D ;

一般的には次のように記述する。ハザードの要因としては、RAW, WAW, CONFLICT (リソース競合) が指定できる。

【 0 0 2 8 】

Stall Stage (ハザード要因、対象とするリソース)

=ストールするステージ;

ストール定義の記述があった場合は、シミュレータは以下に示すアルゴリズムでパイプライン動作を行う。

【 0 0 2 9 】

(シミュレータのストール制御アルゴリズムの一例)

S : ストールするステージ、R : 対象となるリソース、T I : R を利用する先行命令とする。命令が現サイクルで S ステージにあるとき、ハザード要因が R A W のとき、T I が R に書き込むべき値がその時点で利用可能でないならストールする。ハザード要因が W A W のとき、T I が R への書き込みを終えてなければストールする。ハザード要因が C O N F L I C T のとき、T I が R へアクセス中ならストールする。上記以外はストールしない。

【 0 0 3 0 】

ここで、先行命令がレジスタに書き込む値が利用可能かどうかは、先行命令のバイパスとレジスタ書き込みステージの定義により決まる。それについては次に説明する。このようにして定義された A L U 命令のパイプライン仕様記述を以下に示す。

【 0 0 3 1 】

(A L U 命令のパイプライン仕様記述例)

／／A L U パイプライン

／／R A W ハザードが検出された時は、D ステージでストールする。

【 0 0 3 2 】

／／E ステージで A L U 演算を行う。レイテンシは 1 である。

【 0 0 3 3 】

／／E または M ステージで A L U 演算結果をフォワードする。

【 0 0 3 4 】

／／W ステージでレジスタに書き込む。

【 0 0 3 5 】

```

ALUPipe::ALUPipe ()
{
    name = "ALU_Pipeline";
    Stages = F + D + E + M + W;
    //リソースとデータフローの関係
    REG.Read (D);
    ALU.Access (E);
    ALU.Forwarding (E + M);
    REG.WriteBack (W);
    //ハザードによりストールさせるステージ
    StallStage (RAW, REG) = D;
}

```

次に、ロード命令に関するパイプライン仕様記述を例示する。ロード命令はMステージで命令をロードしバイパスする。そのため、直後のデータ依存のある命令は1サイクルストールする。この動作を実現するためのパイプライン仕様記述を以下に示す。

【0036】

(ロードパイプラインの仕様記述例)

//ロードパイプライン

//RAWハザードが検出された時は、Dステージでストールする。

【0037】

//EステージでALU演算（アドレス計算）を行う。

【0038】

//Mステージでデータのロードに成功するとMステージでフォワードする。

【0039】

//Wステージでレジスタに書き込む。

【0040】

```

LDPipe::LDPipe ()
{

```



```

name = "Load_Pipeline";
Stages = F + D + E + M + W;
//リソースとデータフローの関係
REG. Read (D);
ALU. Access (E);
MMU. Load (M);
REG. WriteBack (W);
//ハザードによりストールさせるステージ
StallStage (RAW, REG) = D;
}

```

次に、以下に示すような命令列を考える。

【0041】

```

lw $1. ($10);
and $2. $1;

```

and命令は上述したALUのパイプラインの定義に従うものとし、lwは上述したロードパイプラインの定義に従うものとする。この命令列は以下のようなパイプライン動作をする。小文字はストールを表す。

【0042】

```

lw $1. ($10) F D E M W          ロードパイプライン
and $2. $1      F d D E M W ALUパイプライン
                ↑  ↑

```

(a) (b)

時刻 (a) でDステージに来たので、RAWハザードをチェックする。先行のlwとの間にRAW依存関係があるので、lwの結果が利用可能かどうかを調べる。lwは時刻 (a) でEステージにいるが、ロードパイプラインはForwardingがMステージから始まるのでまだ結果利用可能ではない。そのため、and命令は (a) ではストールする。次の時刻 (b) でもand命令はDステージなのでRAWハザードをチェックする。同じくlwの間にRAW依存関係があるのでlwの結果が利用可能かどうかを調べる。今回はlwがMステージを実行

することになっているので 1 w の結果が利用可能であり、ハザードは検出されず
a n d 命令はストールしない。

【 0 0 4 3 】

上述したロードパイプラインの定義において、以下に示す行を無効にすると、
ロードパイプラインの結果利用可能ステージが W ステージになり、同じ例では a
n d 命令が D ステージが 2 サイクルストールし、バイパス機能がない状況を設定
できる。

【 0 0 4 4 】

MMU. F o r w a r d i n g (M) ;

次に、乗算命令のパイプライン仕様を例示する。乗算パイプラインの定義を以
下に示す。

【 0 0 4 5 】

(乗算パイプラインの仕様記述例)

／／乗算命令が従う

／／オペランドの R E G に対する R A W ハザードが検出された時は、

／／D ステージでストールする。

【 0 0 4 6 】

／／MA の演算は E、M ステージで行う。

【 0 0 4 7 】

／／レイテンシは 2

／／H I ／L O への書き込みは、M ステージで行う。

【 0 0 4 8 】

／／M U L D I V に対するリソース競合があってもストールしない

／／先行命令が D I V なら D I V をキャンセルしてそのまま実行

／／先行命令が M U L なら先行の M U L をつぶしてそのまま実行

M U L T P i p e : : M U L T P i p e ()

{

name = "M u l t i p l y _ P i p e l i n e" ;

```

//out-of-order完了を実現するためステージ名を変更
Stages=F+D+C(2)+X;
//リソースとデータフローの関係
REG.Read(D);
MULDIV,.Access(C);
HI,WriteBack(X);
LO,WriteBack(X);
//ハザードによりストールさせるステージ
StallStage(RAW,REG)=D;
//先行mul/divはキャンセルさせる
F.MakeFlush(Mult);
F.MakeFlush(Div);
CompletionOrder=OutOfOrder//Out-of-Order完了。
【0049】
}

```

乗算命令が乗除算器(MULDIV)で実行されている間は、他の命令がALUを実行してもかまわない。しかし乗算パイプラインのMULDIV実行ステージをALU実行スライドと同じEステージにすると、シミュレータの制御ルールにより、MULDIVをアクセスしている間は他の命令はALUをアクセスできず、ストールしてしまう。これを避けるためにCという新しいMULDIV用のステージを定義している。

【0050】

乗算命令を実行中、依存関係のない後続の命令は先に終了する。すなわちOutOfOrder完了であり、これを指定するのが上記乗算パイプラインの定義における、

```

CompletionOrder=OutOfOrder;//Out-of-Order完了。

```

【0051】

の行である。これを無効にすると後続命令はたとえ乗算命令と依存関係がなくても乗算命令が終了するまでストールする。また、同一ステージに複数の命令が移れないという制御ルールから、乗算命令がCステージを終了しない間では、後続のCステージを使う命令（乗算と除算）は手前のDステージでストールする。これは多サイクルステージでレイテンシを調整した場合に、デフォルトではスループットは多サイクルステージのサイクル数以上になることを意味する。スループットをレイテンシ以下にしたい場合は、ステージ並びを増やす方法をとる。リソース競合などを定義しなければスループットは1となるが、スループットを1以外にするには、パイプライン定義中で、`throughput = 3`のように指定する。

【0052】

先行の乗算や除算命令が終了しないうちに次の乗算や除算命令が投入された場合には、先行命令をキャンセルする制御を定義したのが、

`F. MakeFlush (Mult) ;`

`F. MakeFlush (Div) ;`

である。これは自身がFステージの時点でパイプライン中に他の乗算や除算命令がある場合に、それをキャンセルすることを意味する。一般的には次の書式である。

【0053】

ステージ`MakeFlush`（命令カテゴリー）

最後に、分岐命令のパイプライン仕様を例示する。常に分岐不成立として先読みし、分岐命令がEステージで条件を判定し、分岐を行うときはその時点でのFステージとDステージにある命令をキャンセルするというパイプライン仕様記述を以下に示す。

【0054】

（分岐パイプラインの仕様記述例）

／／分岐パイプライン

／／分岐／ジャンプ、リピート、リターン命令が従う

／／RAWハザードが検出された時は、Dでストールする。

【0055】

／／EステージでALU演算（条件判定）と、PCの上書きを行う。

【0056】

BRPipe::BRPipe ()

{

name = "Branch_Pipeline";

Stages = F + D + E + M + W;

PredictedValue = NotTaken; ／／常に分岐不成立と予測

／／リソースとデータフローの関係

REG.Read (D);

ALU.Access (E);

LP.Read (D);

EPC.Read (D)

／／ハザードによりストールさせるステージ

StallStage (RAW, REG) = D;

／／分岐予測がはずれたとき（分岐命令がEステージでtakenとなったとき）

／／F、Dステージの命令を次のサイクルへ移る時にフラッシュする。

【0057】

E.setControl (&brHzd);

}

以下の記述文で常に分岐不成立の予測を指定している。

【0058】

PredictedValue = NotTaken; ／／常に分岐不成立と予測

PredictedValueにはTakenとNotTakenだけが指定できる。現状では複雑な分岐予測には対応できない。

【0059】

E. setControl (&brHzd) ;

はEステージで分岐予測とそれに伴う動作を行うことを意味している。brHzdは分岐予測の結果を返すクラスオブジェクトである。

【0060】

(分岐予測時のパイプラインの仕様記述例)

```
void BRPipe::ActionBranchpredict
{
  //分岐予測が外れた。
```

【0061】

```
if (brHzd.BranchPredictResult (PredictedValue, PC) == false)
  //F、Dステージの命令をフラッシュ
  F.Flush ();
  D.Flush ();
}
```

Eステージで実行される動作の実体は、上述したように記述される。分岐予測が外れた場合には、F、Dステージの命令を取り出してキャンセルしている。このように、分岐予測の方向、分岐予測を行うステージ、およびそのときのキャンセルすべき命令の指定は容易に行うことができる。

【0062】

本実施形態では、パイプライン仕様記述としてC++の関数で記述例を示したが、必要なデータの入力を要求する専用のユーザインタフェースを用意して、得られたデータをC++関数等に変換することは容易に実現可能である。

【0063】

このように、上記実施形態においては、計算機が解釈可能に記述された同一のパイプライン仕様を元に、RTL実装、パイプラインシミュレータの生成、検証プログラムの生成が行なわれるため、パイプライン仕様の食い違いによる誤りを防ぐことができる。また、検証プログラムの生成はパイプライン仕様から計算機

により自動生成されるため、仕様が正しく記述されていれば検証洩れを防止することができる。さらに、パイプライン仕様を元に人手で実装する部分はRTL記述であるため、RTLシミュレーションの結果とパイプラインシミュレーションの結果とを比較してシミュレーション結果が合わない場合には、RTL実装に問題があると判断することができ、バグの原因を究明する効率が向上する。また、性能向上の目的等でマイクロアーキテクチャ仕様の変更が行なわれた場合でも、容易に対応することができる。

【0064】

図2はこの発明の一実施形態に係るマイクロプロセッサの設計検証装置の構成を示す図である。図2において、この実施形態のマイクロプロセッサの設計検証装置は、図1に示す設計検証手順にしたがってマイクロプロセッサのパイプライン動作の検証を行うものであり、前述したように計算機が解釈可能な記述言語によって記述されたパイプライン仕様を入力するパイプライン仕様入力手段1と、シミュレータのソースプログラムを入力するシミュレータのソースプログラム入力手段2と、パイプライン仕様入力手段1により入力されたパイプライン仕様を保持するパイプライン仕様保持手段3と、入力されたパイプライン仕様ならびにシミュレータのソースプログラムに基づいてパイプラインシミュレータを生成するパイプラインシミュレータ生成手段4と、パイプラインシミュレータ生成手段4により生成されたパイプラインシミュレータを保持するパイプラインシミュレータ保持手段5と、パイプラインシミュレータ生成手段4により生成されたパイプラインシミュレータ又はパイプラインシミュレータ保持手段5で保持されたパイプラインシミュレータに基づいてパイプラインシミュレータを実行するパイプラインシミュレータ実行手段6と、パイプライン仕様保持手段3に保持されたパイプライン仕様に基づいてパイプラインライン動作モデル、検証項目を生成した後検証プログラムを生成する検証プログラム生成手段7と、検証プログラム生成手段7によって生成された検証プログラムを保持する検証プログラム保持手段8と、RTLデータを入力するRTLデータ入力手段9と、RTLデータ入力手段9により入力されたRTLデータを保持するRTLデータ保持手段10と、RTL保持手段10に保持されたRTLデータをに基づいてRTLシミ

ュレーションを実行するRTLシミュレーション実行手段11と、RTLシミュレーション実行手段11により実行されたRTLシミュレーションの結果とパイプラインシミュレータ実行手段6により実行されたシミュレーション結果を比較して、パイプライン動作の正否(PASS/FAIL)を判定するシミュレーション結果比較手段12を備えて構成されている。なお、パイプラインシミュレータ保持手段5は、パイプラインシミュレータ生成手段4により生成されたパイプラインシミュレータを再利用する際には保持したパイプラインシミュレータをパイプラインシミュレータ実行手段6に与えるので、パイプラインシミュレータ生成手段4によりパイプラインシミュレータを毎回生成する場合には、パイプラインシミュレータ保持手段5は不要となる。

【0065】

このような構成において、計算機で解釈可能な記述言語で記述されたパイプライン仕様に基づいて生成された検証プログラムを用いてマイクロプロセッサのRTL記述のシミュレーションを行なうと同時に、同一の検証プログラムを用いてパイプライン・シミュレータを実行し、両者の結果を機械的に比較することにより洩れなくかつ効率よくマイクロプロセッサのRTL記述の論理検証を行なうようにしている。

【0066】

このような実施形態においては、上記実施形態と同様の効果を得ることが可能である。

【0067】

【発明の効果】

以上説明したように、この発明によれば、計算機が解釈可能に記述されたパイプライン仕様を共通に入力して、計算機により検証項目を自動生成し、RTLシミュレーションならびにパイプラインシミュレーションを行うようにしたので、充分かつ正確にパイプラインライン動作を効率よく検証することが可能となる。

【図面の簡単な説明】

【図1】

この発明の一実施形態に係るマイクロプロセッサの設計検証方法の手順を示す

フローチャートである。

【図 2】

この発明の一実施形態に係るマイクロプロセッサの設計検証装置の構成を示す図である。

【図 3】

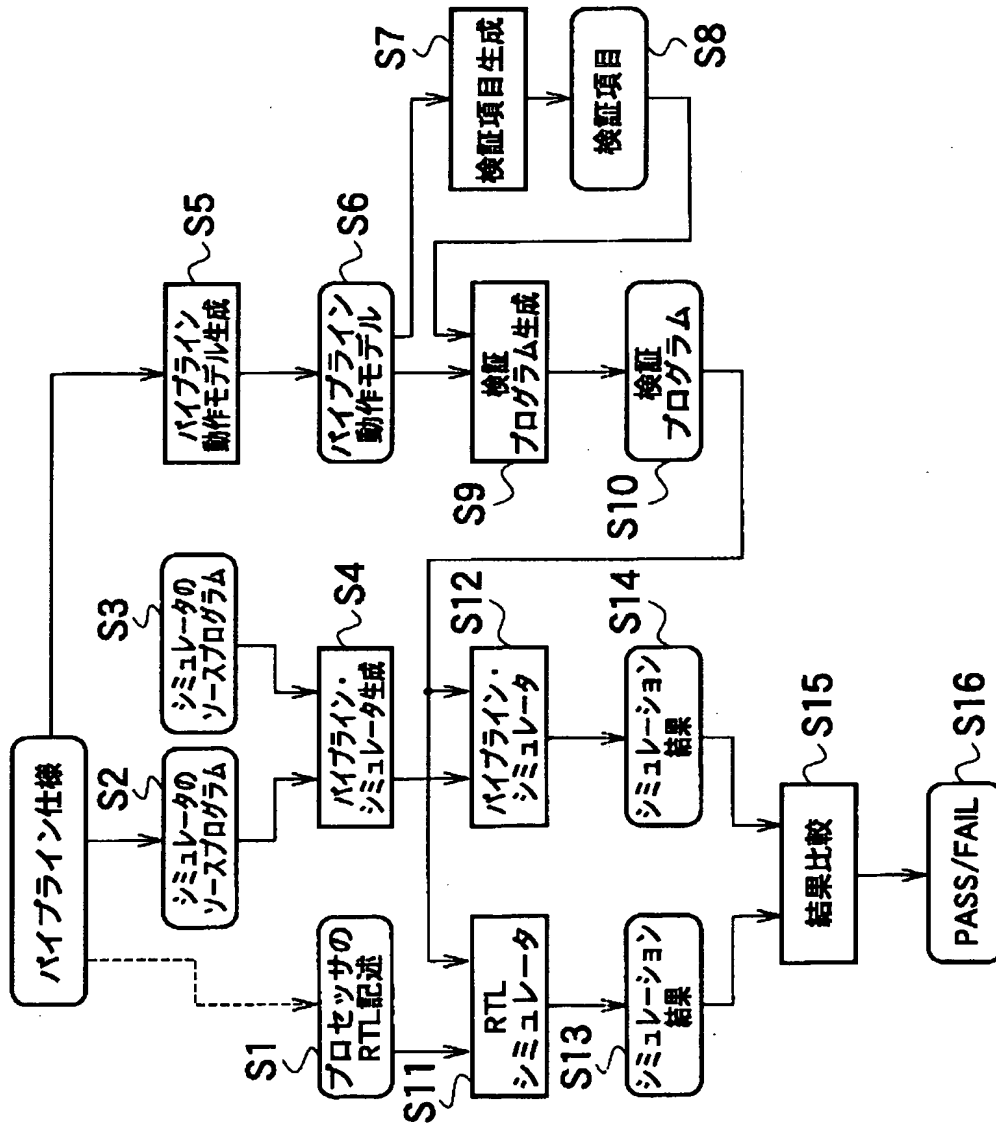
従来のマイクロプロセッサの設計検証方法の手順を示すフローチャートである。

【符号の説明】

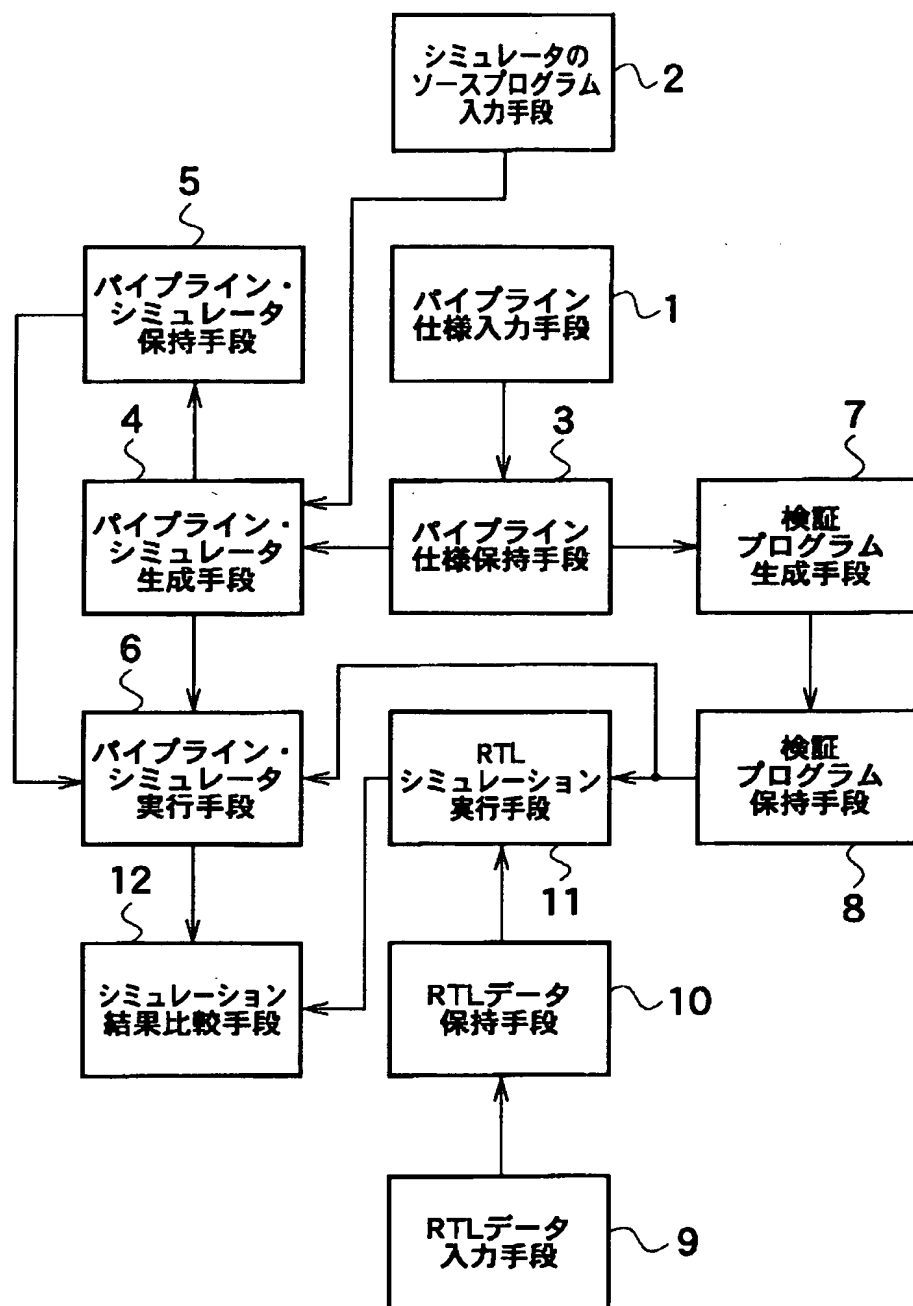
- 1 パイプライン仕様入力手段
- 2 シミュレータのソースプログラム入力手段
- 3 パイプライン仕様保持手段
- 4 パイプラインシミュレータ生成手段
- 5 パイプラインシミュレータ保持手段
- 6 パイプラインシミュレータ実行手段
- 7 検証プログラム生成手段
- 8 検証プログラム保持手段
- 9 RTLデータ入力手段
- 10 RTLデータ保持手段
- 11 RTLシミュレーション実行手段
- 12 シミュレーション結果比較手段

【書類名】 図面

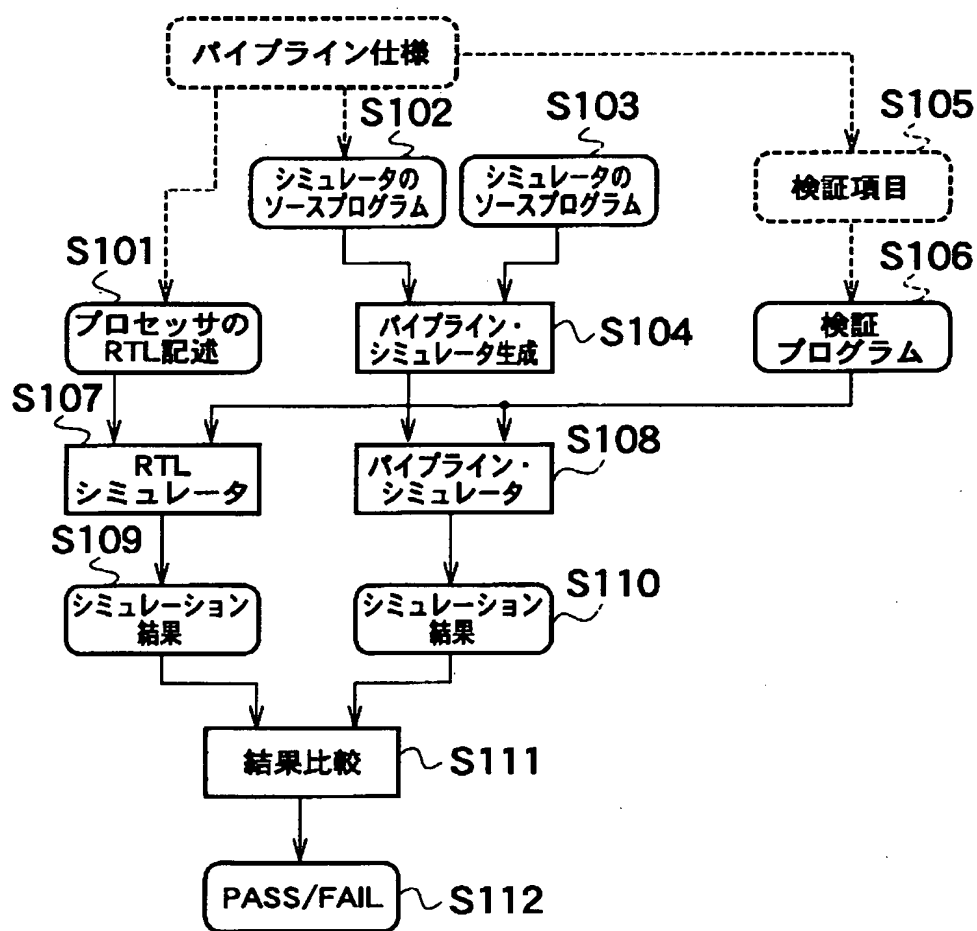
【図 1】



【图 2】



【図 3】



【書類名】 要約書

【要約】

【課題】 この発明は、マイクロプロセッサのパイプライン動作を正確かつ十分に効率よく検証することを課題とする。

【解決手段】 この発明は、計算機が解釈可能に記述されたパイプライン仕様に基づいて、パイプラインシミュレータ、検証プログラムを生成し、検証プログラムと R T L 記述に基づいて実行された R T L 記述のシミュレーションの結果と、検証プログラムとパイプラインシミュレータに基づいて実行されたパイプラインシミュレーションの結果に基づいてパイプライン動作を検証するように構成される。

【選択図】 図 1

出 願 人 履 歴 情 報

識別番号 [000003078]

1. 変更年月日	1990年 8月22日
[変更理由]	新規登録
住 所	神奈川県川崎市幸区堀川町72番地
氏 名	株式会社東芝